Preprint of a paper which appeared in the proceedings of
"RoboCup International Symposium 2008" (Suzhou, China, July 15-18 2008)

Mutual Localization in a Team of Autonomous Robots using Acoustic Robot Detection

David Becker and Max Risler

Technische Universität Darmstadt
Simulation, Systems Optimization, and Robotics Group
Hochschulstr. 10, 64289 Darmstadt, Germany
`{becker|risler}@sim.informatik.tu-darmstadt.de`

Abstract. In order to improve self-localization accuracy we are exploring ways of mutual localization in a team of autonomous robots. Detecting team mates visually usually leads to inaccurate bearings and only rough distance estimates. Also, visually *identifying* teammates is not possible. Therefore we are investigating methods of gaining relative position information *acoustically* in a team of robots.
The technique introduced in this paper is a variant of code-multiplexed communication (CDMA, code division multiple access). In a CDMA system, several receivers and senders can communicate at the same time, using the same carrier frequency. Well-known examples of CDMA systems include wireless computer networks and the Global Positioning System, GPS. While these systems use electro-magnetic waves, we will try to adopt the CDMA principle towards using *acoustic* pattern recognition, enabling robots to calculate distances and bearings to each other.
First, we explain the general idea of cross-correlation functions and appropriate signal pattern generation. We will further explain the importance of synchronized clocks and discuss the problems arising from clock drifts.
Finally, we describe an implementation using the Aibo ERS-7 as platform and briefly state basic results, including measurement accuracy and a runtime estimate. We will briefly discuss acoustic localization in the specific scenario of a RoboCup soccer game.

1 Introduction

Knowing exact relative positions of team mates in a team of autonomous robots can be a valuable information. For example, self-localization accuracy can be improved by including the team mates position beliefs, together with according measurements.

A typical situation in a RoboCup soccer game is a player, who tries to grab and shoot the ball while being attacked by opponent players. Usually, this leads to a bad localization accuracy, because the player is unknowingly being pushed without seeing any landmarks. A well localized team mate can support this player by localizing him and sending him a position belief via wireless communication.

Other applications of knowing relative team mate positions are several issues of tactical game play behavior, and special applications like passing.

Hereby we present an approach of measuring distances and bearings acoustically, using a speaker and a pair of microphones. Besides, the techniques described are platform independent. Measuring distances requires each robot to have a speaker and a microphone installed. Measuring bearings additionally requires a second microphone installed.

Measuring distances is done by measuring the time the sound travels from robot "A" to robot "B". If the sending of signals follows an exact timetable, both A and B know the point in time when A sends. (This usually requires the robots to have synchronized clocks, see section 3.2.) The main challenge for B is now to detect the *exact* point in time when A's signal reaches his microphone, even under noisy conditions. This signal detection technique will be the main topic of this paper. Our approach will allow several robots of a team to send their signals continuously and simultaneously.

While the calculation of bearings is inspired by the human sense of hearing, humans usually are unable to measure distances acoustically. However, there exist everyday life examples, like estimating the distance to a thunderstorm by counting the seconds between lightning and thunder.

Note, that the frequencies used in our approach are *human audible*, in a range between 500Hz and 1000Hz. This might be a downside of the approach when used in a quiet surrounding. The acoustic schemes might sound unpleasant for the human ear. However, the sound volume can be adapted to the level of noise in the environment.

2 State of Research

The *transmission of information* using acoustic signals has been investigated and also successfully used in a RoboCup competition (see [1] and [2]). Gaining *position information* acoustically in a team of robots is, however, a quite unexplored field of research.

We hereby discuss a wireless CDMA communication system based on acoustic signals. The approach mainly deals with pattern recognition using cross correlation functions. A lot of research has already been done regarding wireless CDMA communication (see [3]) and cross correlation functions in general. Well-known applications that use this technique include wireless network standards like WiFi, and also the Global Positioning System (GPS) [4]. Note, that all of these applications are based on electro-magnetic waves, while we aim to use acoustic waves.

There has been some research about gaining position information acoustically. See [5] for an approach for measuring distances between a single sender and a receiver using cross correlation. This is very similar to our approach, while we try to set up such a system for teams of autonomous robots.

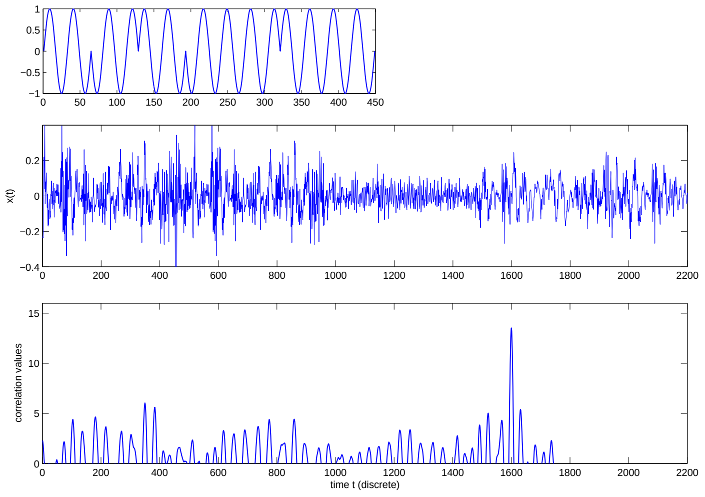

Fig. 1. Top: An example signal pattern p, Middle: A noisy input signal $x(t)$ containing a single pattern p, Bottom: The discrete correlation function $\rho_{x,p}(t)$ (positive range only), which correctly "reveals" the position of the signal pattern: $t_p = 1600$.

2.1 Cross-Correlation-Functions

In the following the process of detecting a signal using cross correlation is described: A specific signal pattern is assigned to each sender, and all receivers know these patterns. After a sender "A" has sent his signal pattern, a receiver "B" calculates the cross correlation function of his input signal and A's specific signal pattern. The cross correlation function $\rho(t)$ of the discrete input signal $x(t)$ and A's discrete signal pattern $p(\tau)$ of length n is a similarity measure, given as

$$\rho_{x,p}(t) = \sum_{i=0}^{n} x(t+i) \cdot p(i).$$

B now assumes, that A's signal arrived at time $t_A = \operatorname{argmax}_t \rho_{x,p}(t)$. Thus, the highest peak, reflecting the position of highest similarity, is chosen. To achieve reliable measurements, this peak should be clearly higher than any other correlation value. Fig 1 gives a simple example.

2.2 Code Generation

The selection of appropriate signal patterns is a crucial issue. Since several senders will send out their signal simultaneously, we have to make sure, that the receivers will not confuse the different patterns.

A signal pattern follows a binary code sequence, usually written as a sequence of ones and zeros. The elements of a binary code sequence are called "chips" instead of "bits", since they are not part of any transmitted data. We give two 15-chip example code sequences here:

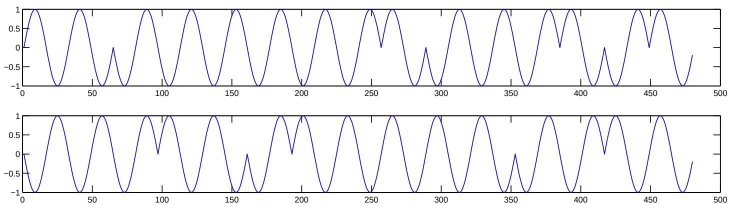

Fig. 2. Signal patterns of k_1 (top) and k_2 (bottom).

$$k_1 = (0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0),$$

$$k_2 = (1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0).$$

Fig 2 shows their corresponding signal patterns. As you can see, chips are modulated here using phase jumps.

Kasami [6] and Gold [7] gave algorithms to create sets of binary sequences with the following properties:

1. Every sequence of the set correlates badly with *any shifted copy* of itself.
2. Any two sequences of the set correlate badly with any shifted or non-shifted copy of each other.

"Correlating badly" means, that absolute correlation values are low. The process of correlating a function with itself is called "auto-correlation". Of course, the auto-correlation value of the *unshifted* sequence is maximal, reflecting maximum similarity, i.e., identity.

The two example code sequences, k_1 and k_2, are part of the same Kasami set of sequences. Their corresponding signal patterns are denoted with p_{k_1} and p_{k_2}. Fig 3 shows their auto-correlation functions and their cross-correlation function. As you can see, the two signal patterns indeed have good auto- and cross-correlation properties: The unshifted auto-correlation peak is at approx. 240, while all other correlation values are < 100. Longer code sequences yield even better correlation properties.

3 Acoustic Robot Detection

In this section we adopt the general strategies of cross-correlation functions and code generation towards using acoustic robot detection in a team of autonomous robots.

3.1 Codes and Modulation

Kasami [6] code sequences have very good auto- and cross-correlation properties, reaching the Welch Lower Bound [8]. A Kasami set consists of $2^{\frac{n}{2}} + 1$ sequences, with n even. Each sequence has length $2^n - 1$. For $n = 4$, a set consists of five

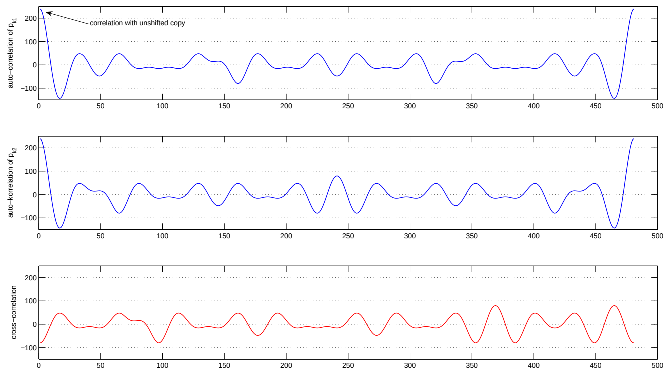

Fig. 3. Auto-correlation functions of p_{k_1} (top) and p_{k_2} (middle), and their cross-correlation function (bottom).

15-chip-sequences. (k_1 and k_2, as introduced in Section 2.2, are part of a Kasami set of $n = 4$.) $n = 6$ yields a set of 9 sequences, with each of length 63. If more codes are required, one could either make codes longer or use Gold codes instead, which have slightly worse correlation properties while enabling much larger sequence sets.

Next, the process of *modulation* is discussed. The previous section already showed how chips can be modulated using phase jumps. This approach yields optimal results: The correlation value of different chips in two signals is the negative of the correlation value of two identical chips. This works well for electro-magnetic waves. However, phase jumps in acoustic waves are difficult to produce accurately with an ordinary loudspeaker. Also, an acoustic waveform, when propagating through the air, will flatten in the regions of phase jumps. (Recall, that acoustic waves are pressure gradients.)

A second approach is to use frequency changes instead of phase jumps. Therefore, we use different frequencies for 0-chips and 1-chips. Obviously, 0-chips and 1-chips must have equal signal lengths, so using a higher frequency requires more sine cycles. The downside of this approach is, that the correlation value of two different chips is now zero, causing a bad overall correlation accuracy.

For a third approach, we return to the idea of phase jumps. If we use more than one sine cycle per chip, the effect of flattening around the phase jump declines. Instead of forcing explicit phase jumps, the waveform can be smoothed out by using short pauses of e.g. a half sine cycle. This approach almost yields the accuracy of the first approach.

The three different modulations of the sequence $k_3 = (1, 0, 1, 1, 0, 0, 0)$ are shown in Fig. 4.

When measuring distances while sending signal patterns continuously, the *length* of the code sequence is of importance. The measured distance is ambiguous since we use finite code lengths. To achieve definite measures, we have to make sure that a signal pattern, when propagating as an electro-magnetic or acoustic waveform, is longer than the maximal possible distance between any receiver

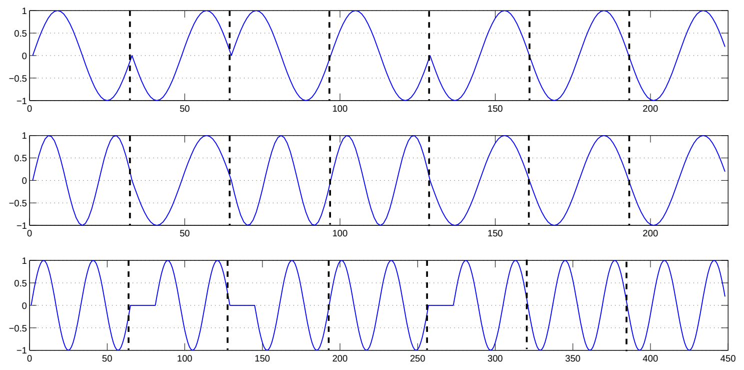
Fig. 4. Three approaches of code modulation: Using phase jumps (top), using frequency jumps (middle), using phase jumps with pauses (bottom)

and sender. For example, assume the distance between two robots never exceeds 20 meters. Using acoustic waves, the minimum pattern length is

$$\chi_{min} = \frac{d_{max}}{c} = \frac{20m}{343\frac{m}{s}} \cong 58ms.$$

3.2 Clock Drift, Clock Synchronization and Distance Calculation

To measure distances, the robots clocks must run synchronized with high accuracy. For example, a clock error of $0.03ms$ equals a distance error of approx. $1cm$.

The clocks usually installed in PCs or mobile robots, respectively, are quartz clocks with very limited accuracy in terms of distance measuring. For example, a clock drift of one minute per month equals a drift of approx. $28ms$ per 20 minutes, which is a measured distance error of more than $9m$.

Hence, a single initial synchronization of the robots' clocks is insufficient. The clocks have to be re-synchronized repeatedly.

Fortunately, there exists an elegant way of synchronizing a pair of robots permanently. Because every sender in our scenario is also a receiver, we always have *two measures* of the *same* actual distance. As a result of the clock drift, these two measures will diverge. Taking the arithmetic average of the two measured distances gives us the *actual* distance.

Fig. 5 gives an example of a mutual distance calculation and synchronization between two robots R1 and R2. Initially, the two robots clocks run asynchronously. Each of the robots was switched on at local time 0. Each robot can determine the time when he sends or receives a signal (t_{S1}, t_{R1}, t_{S2}, t_{R2}). Assume, it takes time t_d for the acoustic waveform to travel from R1 to R2. t_d can be determined as follows: R1 sends $(t_{R1} - t_{S1})$ to R2 via wireless LAN. Then R2 can calculate t_d using the equation

$$t_d = \frac{1}{2}[(t_{R2} - t_{S2}) + (t_{R1} - t_{S1})].$$

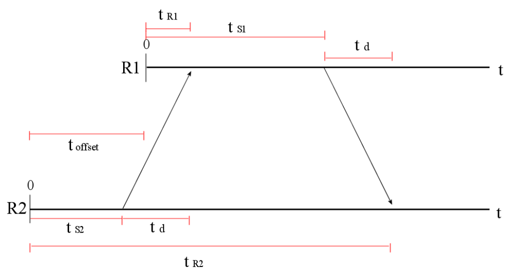

Fig. 5. Time view of the mutual localization and synchronization process.

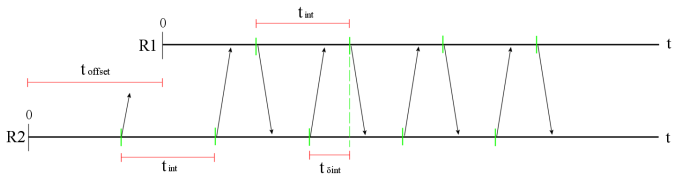

Fig. 6. Time overview: the green lines represent time interval edges.

(R1 calculates t_d equivalently.)

R2 now also knows the local time when R1 sent out the signal. Since the sending of signals is done at regular time intervals t_{int}, R2 also knows the next sending times of R1 in *local time*:

$$t_{S1,i} = t_{R2} - t_d + i \cdot t_{int}, \text{ with } i = 0, 1, 2, ...$$

Therefore, R2 may calculate the distance to R1 only by having t_{R2} in the upcoming intervals.

While this synchronization procedure does not yield the main system time offset t_{offset}, it gives the relative time offset *between two adjacent time interval edges* of R1 and R2, $t_{\delta int}$. Note, that this suffices for measuring t_d. See Fig. 6 for an overview.

Altogether, a pair of robots can synchronize their clocks whenever they detect their signals coevally. For the meantime, they use the relative clock offset calculated at the last synchronization interval to perform a regular distance calculation.

Note, that considering the *relative* clock drift suffices here. There is no need to synchronize the clocks with civil time.

3.3 Bearing Calculation

For the calculation of bearings, the incoming signals from both microphones are compared. The cross-correlation function is evaluated for each of the signals, yielding a distance difference ds. The bearing angle is then given as

$$\alpha = \arcsin(\frac{ds}{b}),$$

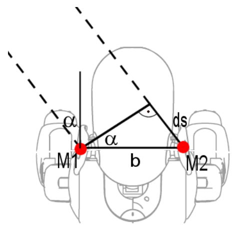

Fig. 7. Calculating the bearing angle α. M_1 and M_2 are the two microphones and b the distance between them.

with b being the distance between the two microphones, see Fig. 7.

Note, that calculating bearings requires a high measurement accuracy for ds, see Section 4.3.

4 Implementation and Results

The four-legged Aibo ERS-7 was used as platform. The Aibo has a small speaker installed in its chest, and one microphone in each ear. The digital sound format we use for both input and output is 16bit at 16kHz sampling rate, which is the highest sound quality the Aibo offers. A RoboCup soccer game usually is a very noisy environment. When walking, however, the majority of noise comes from the Aibo itself.

In signal processing, the mean ratio of signal strength and noise is called *Signal to Noise Ratio (SNR)*. In a RoboCup soccer game, the SNR can be as low as 1:20.

4.1 Selecting Frequency and Code Length

Selecting an appropriate transmitting frequency is finding a trade-off between several issues, of which the most important ones are listed here:

– First of all, we need a high signal strength. The Aibos speaker is much louder at certain frequencies.
– Lower-frequency waves have better propagation properties, as they are propagate more linearly through the air than higher-frequency waves, which are more vulnerable to reflection and diffraction.
– Higher-frequency waves lead to narrower peaks in the cross-correlation function, resulting in a higher accuracy.

Experiments show, that frequencies in the range from $500Hz$ to $800Hz$ provide the best results.

Since there are five players in a four-legged-team, we use a Kasami-set of $n = 6$ yielding 9 code sequences, each of length 63. Using two full sine cycles per chip and a frequency of $f_0 = 750Hz$ gives a pattern of length

$$\chi = \frac{2}{f_0} \cdot 63 = 168ms.$$

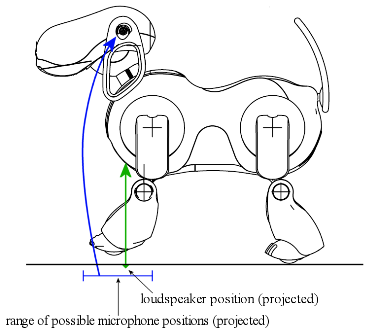
loudspeaker position (projected)
range of possible microphone positions (projected)

Fig. 8. Loudspeaker and microphone positions, projected onto the ground. The projected microphone position depends on the position of the head.

The corresponding acoustic wave has a length of $343\frac{m}{s} \cdot 168ms \approx 57.6m$, which guarantees unambiguous distance measurements in the scenario of a RoboCup soccer game (see Section 3.1).

4.2 Clock Drift

The relative clock drift between two Aibos can be as large as $1ms$ per minute in the worst case, resulting in a distance measurement drift of up to $34cm$ per minute. To keep this error low, the robots need to synchronize their clocks once every two seconds, as described in Section 3.2. This delimits the maximum relative clock drift to $0.033ms$, which is equivalent to $e_{ClockDrift} \approx 1cm$ in distance.

4.3 Distance and Bearing Accuracy

The difference in time of a single sample equals a distance of about $e_{Resolution} = 2cm$ when using a 16kHz sampling rate. This error can be reduced by interpolation of neighboring correlation values.

Another error comes from the fact, that the Aibo's speaker and microphone reference points are not congruent when projected on the ground, depending on the head position. Two Aibos with the same orientation have an estimated maximum error of $e_{Orientation} \leq 4cm$, see Fig. 8. On the contrary, $e_{Orientation} = 0$ for two Aibos facing each other. Since the relative orientation is a priori unknown, we have to bargain for the maximum error.

Diffraction, reflection and interference lead to a third error. The original wave front interferes with diffracted and reflected wave fronts of lower amplitude, yielding small positive phase shifts in the resulting wave. The effect is predominantly noticeable in a signal of a microphone *aiming the opposite direction* of the signal source, because diffractions *at the Aibo's own chassis* seem to make up the biggest portion. Experiments show, that the resulting maximum error of this channel can be estimated by $e_{Interference} \leq 5cm$. The signal of a microphone facing the signal source is almost unaffected by diffraction and interference.

For distance measurements, we use the arithmetic average of the distances to the both microphones yielding a resulting maximum error of $e_{Interference} \cdot 0.5$ here.

The overall *distance* measurement error can be estimated as

$$e_{Distance} \leq e_{ClockDrift} + e_{Resolution} + e_{Orientation} + 0.5 \cdot e_{Interference} \cong 9.5cm.$$

If the signal comes *from the side*, the effect of diffraction and interference yields very inaccurate bearing measurements. We can only make a qualitative statement for the signal source position in that case. The actual error also varies on nearby obstacles like walls or other robots, and therefore can hardly be estimated.

If the signal source is *in front of* the Aibo, within a range of about $\alpha \leq \pm 35°$, the effect of diffraction almost disappears, because the Aibo's chassis does not hamper the wave propagation. However, the resulting bearing measurement error still can be as large as $e_{Bearing} \approx \pm 10°$ due to obstacles.

4.4 Runtime Considerations

Evaluating the correlation function takes the majority of calculation time. At 16kHz sampling rate, a 32ms-frame equals 512 sampling points of sound data.

The evaluation of a *single* correlation value for the sound data portion received in a 32ms-frame needs 512 integer multiplications and 512 integer additions. We can use integer operations, because the signal pattern and the input signal as well are integer values. Note: Since we only look out for a single, distinct maximum peak, the actual, *exact* correlation values are not of any interest.

Finding a players signal correlation peak initially is costly, because we have to calculate all possible correlation values. In our setup, this means 2560 evaluations. Usually, this is too costly to be done in a single 32ms-frame and therefore is split into smaller steps. E.g., if the number of evaluations is limited to 40 for each 32ms-frame, the initial finding of a team mate's signal may take 2 seconds. If we lose a team mate's distance, e.g. when manually replaced by a referee, we know that the distance to that team mate maximally changes by approx. $12m$. Finding this team mate's signal pattern again will take less than half a second.

Once a player's signal was found, i.e. the maximum correlation peak was found, we can use this knowledge for the next time step. Assuming, that the distance to another team mate maximally changes by 20cm during the sending of a signal pattern ($168ms$), the correlation only needs to be done in the corresponding vicinity of the previously found peak position. We refer to this procedure as "distance tracking". Therefore, evaluating only approx. 20 correlation values per 32ms frame is sufficient to track the position of a team mate.

The overall sum of operations to be calculated in a 32ms-frame is given by

- 512 additions and 512 multiplications per evaluation

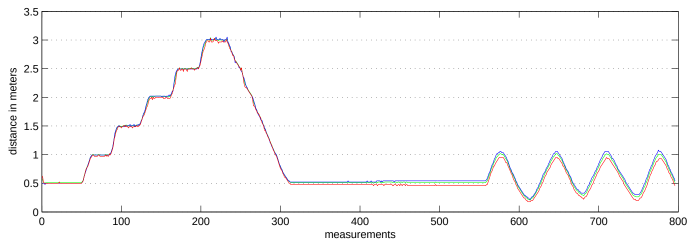

Fig. 9. Measured distances between the two robots. green: Using continuous synchronization; red/blue: using an initial synchronization only.

- 20 evaluations needed for distance tracking
- 2 channels to be evaluated (left and right microphone)
- 4 team mates to be evaluated coevally.

This multiplies up to $81,920$ integer additions and multiplications per 32ms-frame. Note, that additional calculation steps are required for array indexing, memory accesses and increasing of counter variables.

4.5 Experimental Results

In a simple experimental setup, two robots sent their signals simultaneously and continuously. The environment was a laboratory yielding a moderate environmental noise. One robot was manually moved along a measuring tape, enabling us to roughly compare the measurements to the actual distances. The initial distance was 0,5m. Then, one robot was moved away from the other in steps of 50cm, up to a distance of 3m. The robot stood still a few seconds at each of these six positions. After that, the robot was moved back again slowly, with a final distance of again 0,5m. Afterwards, the robot was moved back and forth between approx. 0.25m and 1m.

Fig. 9 shows the measured distances. As you can see, the distances calculated by the robots reflect the coarse of the experiment with a good precision. The green line shows the measured distances using continuous synchronization. The blue and red lines show the measured distances when using only a single initial synchronization. After two minutes, the clock drift already caused an error of about 15cm.

A second experiment was set up in order to test bearing measurement quality. A robot moved around another still-standing robot on a circle of radius 1m. Please recall the denotations in Fig. 7. The values of ds were measured every $22.5°$ on the robots way along the circle. Fig. 10 shows these measurements ds against their corresponding theoretical values. As expected, interferences lead to larger absolute values. As you can see, ds becomes as large as $1.35 \cdot b$ for angles of $\pm 90°$. Obviously, ds can not get larger than b in theory. This tendency is experimentally reproducible. There might be a chance to increase bearing quality by using some correcting function. Overall, bearing measurements are rough, but still can be useful for several purposes.

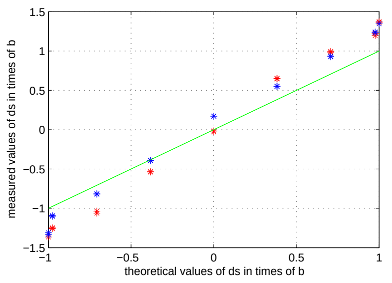

Fig. 10. Measured bearings between the two robots. red: sending the signal from behind, blue: from the front.

5 Conclusions

A robot can measure distances and rough bearings to his team mates several times a second. The technique of using cross-correlation functions turns out to be very cost-efficient, while being robust against noise. However, to achieve good results in *very* noisy environments, the robots have to be equipped with adequate loudspeaker hardware.

The technique discussed is suitable for real-time applications, as computations can easily be spread over several time-frames. The overall process is expandable to larger teams of robots and also easily adoptable to other platforms or different scenarios.

References

1. Chen, S., Siu, M., Vogelgesang, T., et al.: The UNSW RoboCup 2001 sony legged league team report (2001)
2. Burkhard, H.D., Düffert, U., Hoffmann, J., Jüngel, M., Lötzsch, M., Brunn, R., Kallnik, M., Kuntze, N., Kunz, M., Petters, S., Risler, M., v. Stryk, O., Koschmieder, N., Laue, T., Röfer, T., Spiess, K., Cesarz, A., Dahm, I., Hebbel, M., Nowak, W., Ziegler, J.: GermanTeam 2002 available online: http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.
3. R. Kohno, R. Meidan, L.M.: Spread spectrum access methods for wireless communications. IEEE Communication Magazine **41** (1995)
4. Parkinson, B.: Global positioning system: Theory and applications. Technical report, American Institute of Aeronautics and Astronautics, Washington, D.C. (1996)
5. Girod, L.: Development and characterization of an acoustic rangefinder. Technical report, University of Southern California, Los Angeles (2000)
6. Kasami, T.: Weight distribution formula for some class of cyclic codes. Technical Report No. R-285, Univ. of Illinois (1966)
7. Gold, R.: Optimal binary sequences for spread spectrum multiplexing (corresp.). IEEE Transactions on Information Theory **13**(4) (1967)
8. Welch, L.: Lower bounds on the maximum cross correlation of signals. IEEE Transactions on Information Theory **20**(3) (1974)